International Journal of Computer Science Engineering
and Information Technology Research (IJCSEITR)
ISSN(P): 2249-6831; ISSN(E): 2249-7943
Vol. 4, Issue 2, Apr 2014, 167-176
© TJPRC Pvt. Ltd.

PERFORMANCE ENHANCEMENT OFTHE SYSTEM THROUGH CLOUD COMPUTING

SHASHI KANT GUPTA[1] & NEERAJ VIMAL[2]

[1]Research Scholar, Azad Institute of Engineering & Technology, Lucknow, Uttar Pradesh, India

[2]Assistant Professor, Department of IT, Azad Institute of Engineering & Technology, Lucknow, Uttar Pradesh, India

ABSTRACT

The cloud is designated as a three-tier structure, specifically Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS) from low-layer to high-layer. In recent times PaaS is evolving quickly as a software development/ deployment environment for enterprise systems. An enterprise system is commonly contains numerous subsystems to model a complex real business. The subsystems are frequently offered by service on the internet, or PaaS, and the developer of the enterprise system needs to select suitable PaaS providers to develop an effectual system. As the number of PaaS provider will rise, a challenging issue is how to select an appropriate combination of services, or PaaSes, to construct a complex enterprise system. Furthermore, the enterprise system has firm requirements for the quality of service (QoS) as well as a budget restraint. Unfortunately no practical workaround to this complex problem is realized so far in the cloud computing environment.

In this paper we aim to propose a combinatorial auction-based marketplace mechanism for cloud computing services, which permits users to reserve capricious combination of services at wished timeslots, prices and quality of service. Our proposed mechanism supports enterprise users build workflow applications in a cloud computing environment, especially on the platform-as-a-service, where the users need to compose multiple types of services at diverse timeslots.

KEYWORDS: Infrastructure as a Service (IaaS), Software as a Service (SaaS), Platform as a Service (PaaS)

INTRODUCTION

Cloud computing is a distributed computational model over a large pool of shared-virtualized computing resources (e.g., storage, processing power, memory, applications, services, and network bandwidth), where customers are provisioned and de-provisioned recourses as they need. Cloud computing represents a vision of providing computing services as public utilities like water and electricity. The architecture of cloud computing can be split in two: front-end and back-end. The front-end represents cloud customers, organizations, or applications (e.g., web browsers) that use the cloud services. The back-end is a huge net- work of datacenters with many different applications, system programs, and data storage systems. It is metaphorically believed that, cloud service providers (CSPs) have almost infinite computation power and storage capacity. A conceptual framework of cloud computing architecture is illustrated in Figure 1 with its two main parts.

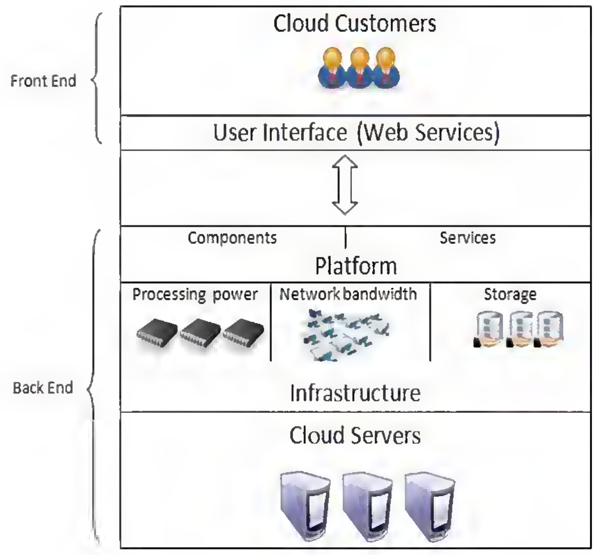

Figure 1: Conceptual Framework for Cloud Computing Architecture

Cloud computing services can be categorized into following:

- Application-as-a-Service (AaaS).
- Platform-as-a-Service (PaaS).
- Infrastructure-as-a-Service (IaaS).

The widely used model of cloud computing services is the AaaS model, in which the customers have access to the applications running on the cloud provider's infrastructure. Google Docs, Google Calendar, and Zoho Writer are known examples of this model. In the PaaS model, the customers can deploy their applications on the provider's infrastructure under condition that these applications are created using tools supported by the provider. The cloud service provider (CSP) hosts a set of software and development tools on its servers to be used by the developers to create their own applications. Google Apps is one of the best known PaaS models. IaaS model enables customers to rent and use the provider's resources (storage, processing, and network).Hence, the customers can deploy any applications including operating systems.

The cloud computing architecture can be deployed under different models:

Public Cloud: The infrastructure of the CSP is publicly accessible by general customers and organizations in exchange for pre-specified fees according to the usage of The CSP's services.

Private Cloud: The cloud infrastructure is dedicated to an organization which may manage the infrastructure or leave this management to a third party.

Hybrid Cloud: The cloud infrastructure is composed of two or more clouds (private or public).The organizations provide and handle some internal and external resources.

Enterprise System

An enterprise system usually consists of multiple subsystems running in parallel and/or sequentially, each of which requires a guaranteed quality of service (QoS) at a unsurprising price. Each enterprise system, or each user's demand, is assumed to be represented by workflow. An example of business workflow is a payroll system. It consists of a payroll calculation task on Java service along with an employee database task on SQL service, followed by reporting task on PDF/Email service as shown in Figure 2.

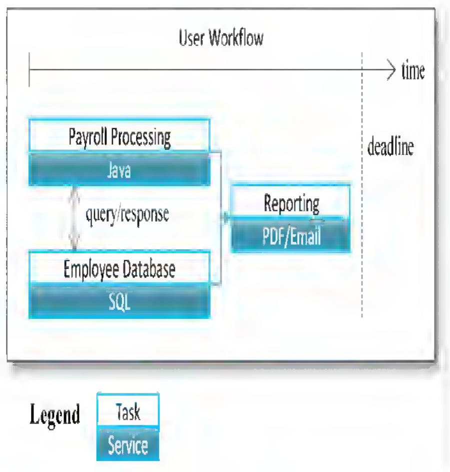

Figure 2: Enterprise System Model

Another example of engineering workflow is a CAE system. It consists of a mesh generation task and a CFD analysis task on a HPC service, controlled by an optimization task on a general-purpose optimization service. Every task needs to reserve the specified type of service within an appropriate timeslots to meet a deadline. The overall cost should also be restricted by the user's total budget. Each task in the workflow is implemented using PaaS; thus, the user needs combination of PaaS services to organize the user's workflow.

Marketplace Model

Figure 3 illustrates an idyllic marketplace model that fits to the enterprise usage and requirements mentioned above. It should support multiple types of services, an application composed of multiple services as a workflow or as co-allocation, price bidding by both providers and users, and a fair outcome satisfying economic efficiency.

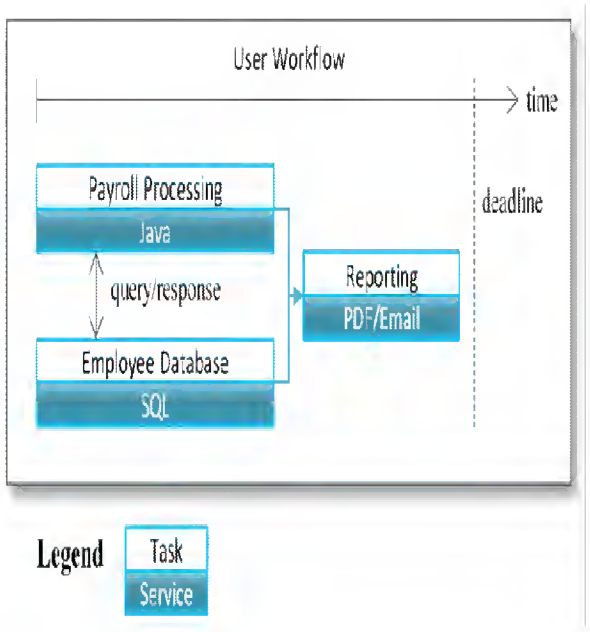

Figure 3: Marketplace Model

LITERATURE REVIEW

- Bellagio by AuYoung et al. [3] is the first combinatorial auction-based marketplace for distributed computing resources. It aims to be a resource discovery and allocation system for distributed computing infrastructures such as PlanetLab [4]. Bellagio employs periodical, combinatorial, single-sided, sealed-bid, second-price auctions. The buyers are end-users, the seller is a computer site, the auctioneer is centralized Bellagio server, and the good is a right to use a resource discovered by SWORD [5].
- Mirage by Chun et al. [6] is another combinatorial auction-based resource management system for Sensor Net test beds. It had been deployed in a real-world test bed and brought practical knowledge about the users' behavior. Mirage employs periodical, combinatorial, single-sided, sealed-bid, first-price auctions. The buyers are end-users, the seller is an agent on behalf of the sensors, the auctioneer is Mirage daemon running on front-end web server, and the good is access to the sensor.
- Tycoon [7] by Lai et al. is a market-based distributed resource allocation system based on proportional share, where the resources are allocated in proportion to the amount of money the user spends. It aims to allow users to differentiate the value of their jobs in a cluster computing environment. Tycoon employs Auction Share algorithm to allocate resources instantly and reliably. The buyers are user agents, the sellers are hosts in the cluster, the auctioneer is a process running on the host, and the good is a right to use resources like CPU cycles. Note that there are multiple independent auctioneers on each host.
- CATNETS by Eymann et al. [8-11] compares the decentralized Catallactic approach with the centralized auction-based approach for resource allocation in Grid computing environment. Although the authors concluded that the decentralized approach fits to the Grid environment in terms of scalability, hereafter only the centralized approach is focused as it is the main concern of this work. The centralized approach employs periodical, combinatorial, double-sided, sealed-bid, K-pricing auctions. CATNETS divides the trading into two layers: a service market and a resource market. In the service market, the buyers are Complex Service Agents on behalf of the end-users, the sellers are Basic Service Agents, and the goods are services like PDF generation service. In the resource market, the buyers are Basic Service Agents, the sellers are Resource Service Agents on behalf of the owners, and the goods are resources like CPU/memory/storage/etc. In both markets, the auctioneer is an independent entity.
- SCDA by Tan et al. [12, 13] proposes an iterative combinatorial exchange for resource allocation in Grid computing environments, which essentially emulates a combinatorial auction by doing single-good auctions repeatedly. It aims to eliminate unnecessary volatility of the market price observed in conventional continuous double auctions. SCDA employs continuous, single-good, double-sided, sealed-bid, K-pricing auctions. The buyers are user agents, the seller is an owner agent, the auctioneer is an independent agent, and the good is an arbitrary kind of computing resource.

PROPOSED FRAMEWORK

We propose a framework named C-Mart (Cloud-Mart) to explore market behavior by means of multi-agent simulations. The overall architecture of C-Mart is illustrated in Figure 4. The Exchange and the agents are two main parts of C-Mart. The exchange has two market instances, namely the forward market and the spot market, independently. Each market mechanism is implemented on the top of MACE [14], which is a Java framework for combinational auction.

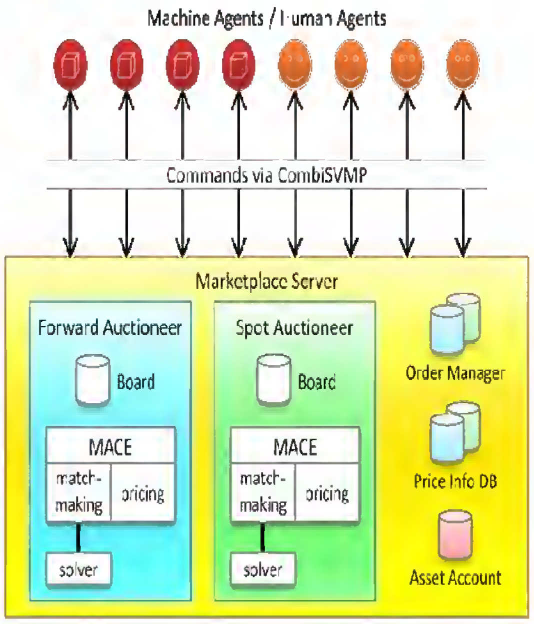

Figure 4: Overview of C-Mart System

The C-Mart system consists of a marketplace server, two auctioneers for the spot and forward markets inside the market place, and a number of participant agents outside the market place. The agent is either a machine agent (i.e. autonomous client software) or a human agent (i.e. an operator with terminal software). The marketplace is designed to deal with both types of agents simultaneously. The agents and the marketplace communicate by a dedicated protocol named CombiSVMP. All the components of C-Mart system are built on Java SE platform. The following sections describes the detailed design and implementation.

Participant Agents

The participant agents simulate the provider and the user of cloud computing services. An unlimited variation of agents can be developed and used with C-Mart as far as they talk CombiSVMP.

Seller Agent

A seller agent stands for a provider of cloud computing services. It is reasonable to assume that the provided quantity and valuation of services are constant, since these services are usually hosted on computational resources in huge datacenters. As such the seller agent implements a timeline to manage his "stock" resources and attempts to sell the "remainder" resources at a constant order price. We assume that one seller provides one service for the sake of simplicity.

Buyer Agent

A buyer agent stands for a user of cloud computing services. As the user's demand changes dynamically in realistic situations, our buyer agent is designed to follow pre-generated demands (i.e. a schedule of workflows). As such the buyer agent implements a timeline to manage his demands and attempts to buy all the services composing the workflows at the specified (also pre-generated) order price.

Scenario Generator

The schedule of workflows fed to the buyer agent is generated by scenario generator and stored in a CSV file.

Timing of interactions with these entities is illustrated in Figure 5. Here the first day of the market place starts with accepting orders from human agents and machine agents. After receiving all orders the marketplace kicks off two auctioneers to execute auctions. Here the first hour ends and the next hour begins. The marketplace keeps open for human orders while waiting for the spot auctioneer to finish. After some ten minutes past, the marketplace closes for human orders and joins the spot auctioneer. Having the result of the spot auction, the marketplace then calls machine agents to make orders. After all the machine agents place their orders, the marketplace again kicks off the spot auctioneer. This is the one round of spot trading and the marketplace repeats it until 21 o'clock.

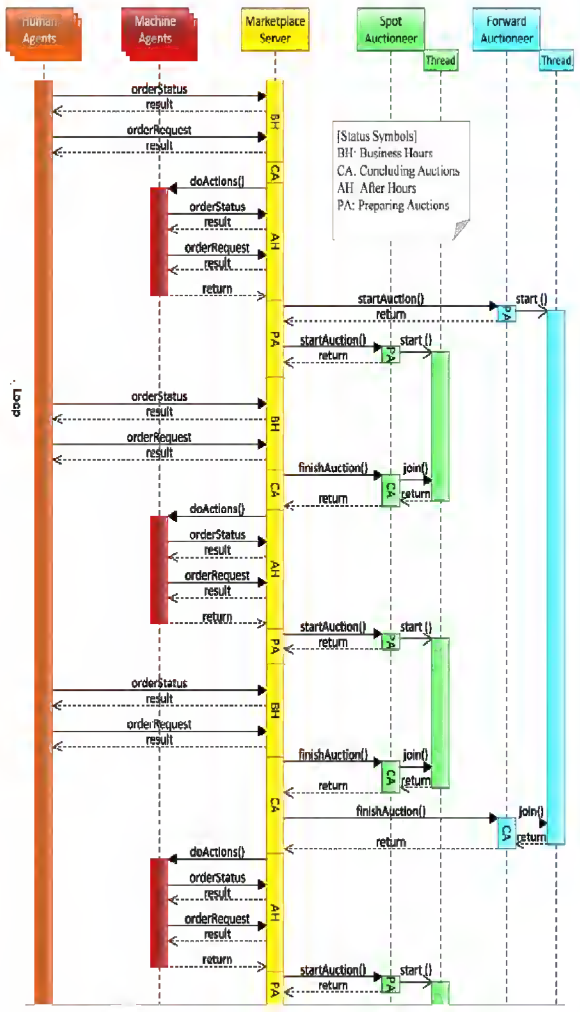

Figure 6: Detailed Sequence Diagram of C-Mart System

Auctioneers

There are two auctioneers within the marketplace server, namely the forward auctioneer and the spot auctioneer, standing for the forward market and the spot market, respectively. As the spot auctioneer is a special case of the forward auctioneer where the time-related parameters are omitted, we focus on the forward auctioneer and describe its design and implementation in this section.

Figure 5 illustrates that the auctioneer has its own thread to run asynchronously with the marketplace server. After the agents finish placing their orders, the marketplace server moves all orders (including un contracted orders of previous session) from the Order Manager to the Board of the auctioneer, and then he starts the auctioneer's thread. Once started, the auctioneer translates the orders from the Board into the MACE [14] framework, and kicks it off to execute matchmaking and pricing. When the MACE framework completed the calculation, the auctioneer scans the outcome and writes it back to the Board. On the other hand, the marketplace server goes asynchronously to the next session, and waits for the auctioneer's thread to terminate. Here the marketplace server may be waiting for the thread to finish. After that the marketplace server moves the outcome from the Board to the Order Manager, as well as the market information (such as the market price and volume) from the Board to the Price Info DB. Finally the outcome and the market information can be retrieved by the agents via the Order Status and the Market Price commands, described in the following section.

The MACE framework embedded in the auctioneer plays two main roles: (1) a driver of combinatorial auction and (2) an executer of pricing scheme. The Winner Determination component of MACE is responsible for computing an allocation, i.e. it implements the mixed integer program. After the computation of the allocation finishes, the prices are computed by the Pricing component of MACE.The bottommost layer of MACE comprises of third party components including an MIP solver. The current implementation of MACE employs JOpt [15] as a wrapper to support multiple MIP engines, namely CPLEX [16] and lp_solve [17]. CPLEX is the state-of-the-art optimization engine for mixed integer programs, whereas lp_solve is an open source alternative implementing branch-and-bound method for solving integer problems. For performance reasons, lp_solve is only used for debugging purpose whereas CPLEX is applied for evaluation of the mechanism.

Protocol

A dedicated protocol, named CombiSVMP (stands for Combinatorial Simple Virtual Market Protocol), is proposed to be designed to exchange information between the marketplace server and the participant agents. Its basic design is derived from SVMP, which is developed for U-Mart v2 system [18]. CombiSVMP extends SVMP to be capable of (1) simultaneous interaction with multiple auctioneers, (2) combinatorial trading of multiple goods and (3) forward trading of multiple timeslots

Followings are the commands defined by CombiSVMP. Most of the commands have AuctioneerName property as a string to specify the auctioneer with whom the agent communicates. Some non-trivial commands and properties are described below.

- The Order Request command is used to place a sell/buy order. It has Order Spec property, which includes arbitrary number of goods and their properties like quantity, earliest timeslot, latest timeslot and total number of timeslot to provide/use the goods.
- The Order Status command is used to inquire the outcome of past orders. It returns Outcome Spec property, which includes two arrays (vectors) representing the quantities and the prices to be sold at each timeslots.
- The Market Price command is used to inquire the historical information of the market. It returns a specified number of histories of Market Price Table's, which includes the market price (i.e. the average sold price per unit per hour) for each good and each timeslot.

Such an exhaustive capability of CombiSVMP allows participant agents to make full use of combinatorial/forward trading features of the proposed marketplace.

CONCLUSIONS

The proposed market mechanism is so generic that any kind of service can be traded equally upon it. In a realistic scenario, however, the marketplace will be used hierarchically. For instance, a bundle of a low-level "storage service" and a high-level "corporate management consulting service" is not likely to be ordered, but that of a "storage service" and another low-level "networking service" are likely to be ordered. There appears a hierarchical structure within the marketplace.

We believe that it will be a keystone for a nation to have control over the cloud computing marketplace. The computing power is now fundamental to any business, science, government, military affairs, and social activities – we cannot live without them – and will be traded beyond the border as a commodity. Therefore, the meta-information generated by the marketplace (i.e. the worldwide flow/price/ demand/supply/etc. of the computing power) can indicate the movement of the world, and the quickest knowledge about it will bring an invaluable advantage to the nation who controls the marketplace.

ACKNOWLEGEMENTS

I take this opportunity to express a deep sense of gratitude towards my guide Asstt. Prof. Neeraj Vimal, Dept. of Information Technology, AIET, Lucknow for providing excellent guidance, encouragement and inspiration throughout the project work. Without his invaluable guidance, this work would never have been a successful one. The perfection that he brings to each and every piece of work that he does always inspired me to do things right at first time. He is a great person and one of the best mentors, I always be thankful to him. I would also like to thank all my classmates for their valuable suggestions and helpful discussions.

I would like to place on record my deep sense of gratitude to Asstt. Prof.& Head of Dept. Madhulika Sharma, Dept. of Computer Science and Engineering, AIET, Lucknow for her generous guidance, help and useful suggestions.

I am extremely thankful to Prof. (Dr.) Shafeeq Ahmad, Director, AIET, Lucknow for providing me infrastructural facilities to work in, without which this work would not have been possible.

I would like to dedicate this thesis to my amazingly loving and supportive parents who have always been with me, no matter where I am.

REFERENCES

1. D. Amrhein, P. Anderson, A. de Andrade, J. Armstrong, E.A. B, R. Bruklis, K. Cameron, R. Cohen, A. Easton, R. Flores, G. Fourcade, T. Freund, B. Hosseinzadeh, W.J. Huie, P. Isom, S. Johnston, R. Kulkarni, A. Kunjunny, T. Lukasik, G. Mazzaferro, C. McClanahan, W. Melo, A. Monroy-Hernandez, D. Nicol, L. Noon, S. Padhy, G. Pfister, T. Plunkett, L. Qian, B. Ramachandran, J. Reed, G. Retana, D. Russell, K. Sankar, A.O. Sanz, W. Sinclair, E. Sliman, P. Stingley, R. Syputa, D. Tidwell, K. Walker, K. Williams, J.M. Willis, Y. Sasaki, E. Windisch, and F. Zappert, Cloud Computing Use Cases White Paper Version 2.0, 2009.

2. S. Weston, D. Green, G. Katsaros, T. Seed, N. Mc Donnell, and F. Scharinger, GridCAE Grid for Computer Aided Engineering, 2009.

3. A. AuYoung, B.N. Chun, A.C. Snoeren, and A. Vahdat, “Resource Allocation in Federated Distributed Computing Infrastructures,” Proceedings of the 1st Workshop on Operating System and Architectural Support for the Ondemand IT Infrastructure, 2004, pp. 1-10.

4. L. Peterson, T. Anderson, D. Culler, and T. Roscoe, “A blueprint for introducing disruptive technology into the Internet,” ACM SIGCOMM Computer Communication Review, vol. 33, Jan. 2003, pp. 59-64.

5. D. Oppenheimer, J. Albrecht, D. Patterson, and A. Vahdat, Scalable wide-area resource discovery, 2004.

6. B.N. Chun, P. Buonadonna, A. AuYoung, D.C. Parkes, J. Shneidman, A.C. Snoeren, and A. Vahdat, “Mirage: A Microeconomic Resource Allocation System for Sensornet Testbeds,” The Second IEEE Workshop on Embedded Networked Sensors, 2005. EmNetS-II., IEEE, , pp. 19-28.

7. K. Lai, L. Rasmusson, E. Adar, L. Zhang, and B.A. Huberman, “Tycoon: An implementation of a distributed, market-based resource allocation system,” Multiagent Grid Syst., vol. 1, 2005, pp. 169-182.

8. T. Eymann, M. Reinicke, W. Streitberger, O. Rana, L. Joita, D. Neumann, B. Schnizler, D. Veit, O. Ardaiz, P. Chacin, I. Chao, F. Freitag, L. Navarro, M. Catalano, M. Gallegati, G. Giulioni, R.C. Schiaffino, and F. Zini, “Catallaxy-based Grid markets,” Multiagent and Grid Systems, vol. 1, Dec. 2005, pp. 297-307.

9. B. Schnizler, D. Neumann, D. Veit, M. Reinicke, and W. Streitberger, Theoretical and Computational Basis for CATNETS - Annual Report Year 1, 2005.

10. D. Veit, G. Buss, B. Schnizler, and D. Neumann, Theoretical and Computational Basis for CATNETS - Annual Report Year 2, 2006

11. D. Veit, G. Buss, B. Schnizler, and D. Neumann, Theoretical and Computational Basis for CATNETS - Annual Report Year 3, 2007.

12. Z. Tan and J.R. Gurd, “Market-based grid resource allocation using a stable continuous double auction,” Grid Computing, 2007 8th IEEE/ACM International Conference on, 2007, pp. 283-290.

13. T. Zhu and J.R. Gurd, “Market-based grid resource allocation using a stable continuous double auction (Ph.D. thesis),” 2007.

14. B. Schnizler, “MACE: A Multi-attribute Combinatorial Exchange,” Negotiation, Auctions, and Market Engineering, 2008, pp. 84-100.

15. “JOpt” www.eecs.harvard.edu/econcs/jopt/.

16. “IBM ILOG CPLEX Optimizer”http://www.ibm.com/software/integration/optimization/cplex-optimizer/.

17. “lp_solve” http://tech.groups.yahoo.com/group/lp_solve/.

18. H. Kita, H. Sato, N. Mori, and I. Ono, “U-Mart system, software for open experiments of artificial market,” Computational Intelligence in Robotics and Automation 2003 Proceedings 2003 IEEE International Symposium on, Ieee, 2003, pp. 1328-1333 vol.3